Dynamic group-based differential evolution using a self-adaptive strategy for global optimization problems

Ming-Feng Han · Shih-Hui Liao · Jyh-Yeong Chang · Chin-Teng Lin

Published online: 1 November 2012

Abstract This paper describes a dynamic group-based differential evolution (GDE) algorithm for global optimization problems. The GDE algorithm provides a generalized evolution process based on two mutation operations to enhance search capability. Initially, all individuals in the population are grouped into a superior group and an inferior group based on their fitness values. The two groups perform different mutation operations. The local mutation model is applied to individuals with better fitness values, i.e., in the superior group, to search for better solutions near the current best position. The global mutation model is applied to the inferior group, which is composed of individuals with lower fitness values, to search for potential solutions. Subsequently, the GDE algorithm employs crossover and selection operations to produce offspring for the next generation. In this paper, an adaptive tuning strategy based on the well-known 1/5th rule is used to dynamically reassign the group size. It is thus helpful to trade off between the exploration ability and the exploitation ability. To validate the performance of the GDE algorithm, 13 numerical benchmark functions are tested. The simulation results indicate that the approach is effective and efficient.

M.-F. Han (✉) · S.-H. Liao · J.-Y. Chang · C.-T. Lin
Institute of Electrical Control Engineering, National Chiao Tung University, 1001 University Road, Hsinchu 300, Taiwan, ROC
e-mail: ming0901@gmail.com

S.-H. Liao
e-mail: liao622@gmail.com

J.-Y. Chang
e-mail: jychang@mail.nctu.edu.tw

C.-T. Lin
e-mail: ctlin@mail.nctu.edu.tw

Keywords Evolutionary algorithm (EA) · Differential evolution (DE) · Adaptive strategy · Optimization

1 Introduction

Evolutionary algorithms (EAs) have become a popular optimization tool for global optimization problems [1–7]. The optimization processes the EAs usually adopt are stochastic search techniques that work with a set of individuals (i.e., solutions) instead of just a single solution, using evolution operators to naturally produce offspring for the next generation. Algorithms such as genetic algorithms (GAs) [8–13], evolutionary programming (EP) [14], evolution strategies (ESs) [15], particle swarm optimization (PSO) [9–11, 16–20] and differential evolution (DE) [21–28] are well-known, effectual and classical evolutionary methods.

Among EAs, differential evolution has interested researchers [29–42] in recent years. The DE algorithm proposed by Storn and Price [21, 22] is an efficient and effective global optimizer in the continuous search domain. It has been shown to perform better than genetic algorithms or particle swarm optimization in several numerical benchmarks [21, 22, 36, 43]. The DE algorithm employs the difference between two randomly selected individuals as the source of random variations for a third individual. It can be applied to difficult optimization problems [21, 22]. However, the DE algorithm may favor the exploitation ability or the exploration ability [31, 32]. To address this problem, Rahnamayan et al. [40] proposed a faster global search and optimization algorithm, called the opposition-based differential evolution (ODE). The ODE employed opposition-based learning to choose better solutions by simultaneously checking the fitness of the opposite solution in the current population. ODE processes can increase the diversity of the population. In [30, 41, 44], the researchers proposed a modified differential evolution (MODE) for an adaptive neural fuzzy network and a locally recurrent neuro-fuzzy system design. MODE provides a cluster-based mutation scheme to prevent the algorithm from being trapped in local optima of the search space. The concept of balancing the exploration and exploitation abilities was proposed by Das et al. [31]. These authors designed a novel mutation operator, the neighborhood-based mutation operation, to handle the problem of stagnation. In their study, they utilized a new mutation strategy with the ring topology of the neighborhood to find potential individuals in the population. Noman and Iba [37] proposed an adaptive crossover operation with local searching (XLS) to increase the exploitation ability of the DE algorithm. The adaptive XLS uses a simple hill-climbing algorithm to adaptively determine the search length for the crossover process. These authors successfully trained the DE algorithm to effectively explore the neighborhood of each individual and locate the global optimum at a minimum solution.

Unlike previous studies, this study develops a new process without depending on other learning algorithms to solve the imbalanced evolution problem. This process employs the inherent properties of the DE algorithm. The process combines two classical mutation strategies instead of a single mutation model. The DE/rand/bin approach has a powerful exploitation ability, and the DE/best/bin approach has an efficient exploration ability. The approach in this study combines the two operations to achieve a balance between the exploration ability and the exploitation ability.

In this study, a group-based DE (GDE) algorithm is proposed for numerical optimization problems. The GDE model is a new process based on two mutation operations. Initially, all individuals in the population are grouped into a superior group and an inferior group, based on their fitness values. The superior group uses the DE/rand mutation operation to search potential solutions and maintain population diversity. The inferior group employs the DE/best mutation model to increase convergence. Subsequently, the crossover and selection operations are employed for offspring production. A self-adaptive strategy based on the 1/5th rule for automatically tuning group size is then applied in the proposed process. Finally, the proposed GDE algorithm is applied to 13 well-known numerical benchmark functions, including unimodal and multimodal function problems. The contributions of this study are summarized as follows:

(1) The proposed GDE algorithm employs the inherent properties of the DE algorithm to solve the evolution imbalance problem. The GDE algorithm combines two mutation operations to balance the exploration ability and the exploitation ability.
(2) A self-adaptive strategy based on the 1/5th rule is proposed to automatically tune the group size without prior user knowledge, improving the robustness of the algorithm.
(3) Thirteen well-known numerical benchmark functions are used to validate the performance of the GDE algorithm. In statistical tests, the GDE algorithm shows significantly better performance than other EAs.

The rest of this paper is organized as follows. Section 2 describes the basic procedure of differential evolution. The GDE flow chart and details of the adaptive group size control strategy are presented in Sect. 3. Simulation results comparing GDE with other evolutionary algorithms are presented in Sect. 4. Concluding remarks are presented in Sect. 5.

2 Differential evolution

This section introduces a complete DE algorithm. The process of the DE algorithm, like other evolutionary algorithms, produces offspring of the next generation by mutation operations, crossover operations and selection operations. Figure 1 shows a standard flow chart of the DE algorithm.

Initially, a population of NP D-dimensional parameter vectors, which represent the candidate solutions (individuals), is generated by a uniformly random process. All individuals and the search space are constrained by the prescribed minimum $\mathbf{X}_{\min} = (x_{1,\min}, x_{2,\min}, \ldots, x_{D,\min})$ and maximum $\mathbf{X}_{\max} = (x_{1,\max}, x_{2,\max}, \ldots, x_{D,\max})$ parameter bounds. A simple representation of the ith individual at the current generation, Gen, is shown as follows:

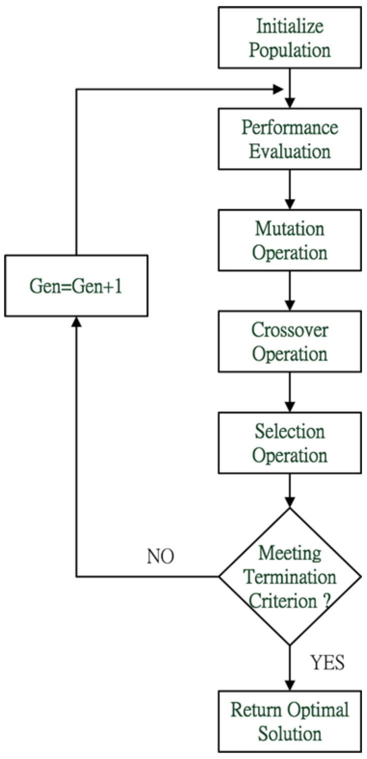


Fig. 1 The flow chart of basic DE algorithm. *Gen* is the generation counter

$$\mathbf{X}_{i,Gen} = (x_{i,1,Gen}, x_{i,2,Gen}, x_{i,3,Gen}, \ldots, x_{i,D-1,Gen}, x_{i,D,Gen}). \quad (1)$$

After the first *NP* individuals are produced, the fitness evaluation process measures the quality of the individuals to calculate the individual performance. The succeeding steps, including mutation, crossover and selection, are described in the following sections.

2.1 Mutation operation

Each individual in the current generation is allowed to breed through mating with other randomly selected individuals from the population. The process randomly selects a parent pool of three individuals to produce an offspring. For each individual $\mathbf{X}_{i,gen}$, $i = 1, 2, \ldots, NP$, where *gen* denotes the current generation and *NP* is population size, three other random individuals, $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$ and $\mathbf{X}_{r3,gen}$ are selected from the population such that $r1$, $r2$ and $r3 \in \{1, 2, \ldots, NP\}$ and $i \neq r1 \neq r2 \neq r3$. In this way, a parent pool of four individuals is formed to produce an offspring. The following are different mutation strategies frequently used in the literature:

$$\text{DE/rand/1:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \quad (2)$$

$$\text{DE/best/1:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F(\mathbf{X}_{r1,gen} - \mathbf{X}_{r2,gen}) \quad (3)$$

DE/target-to-best:

$$\mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{gbest,gen} - \mathbf{X}_{r1,gen}) + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) \quad (4)$$

$$\text{DE/rand/2:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{r1,gen} + F(\mathbf{X}_{r2,gen} - \mathbf{X}_{r3,gen}) + F(\mathbf{X}_{r3,gen} - \mathbf{X}_{r4,gen}) \quad (5)$$

$$\text{DE/best/2:} \quad \mathbf{V}_{i,gen} = \mathbf{X}_{gbest,gen} + F(\mathbf{X}_{r1,gen} - \mathbf{X}_{r2,gen}) + F(\mathbf{X}_{r3,gen} - \mathbf{X}_{r4,gen}) \quad (6)$$

where F is the scaling factor $\in [0, 1]$ and $\mathbf{X}_{gbest,gen}$ is the best-so-far individual (i.e., $\mathbf{X}_{gbest,gen}$ stores the best fitness value in the population up to the current time). The DE algorithm usually employs a different mutation strategy, depending on the problem being solved. The "DE/rand/1" and "DE/rand/2" mutations, with more exploration ability, are suitable for multimodal problems. The "DE/best/1", "DE/best/2" and "DE/target-to-best/1" mutations, which consider the best information in the generation, are more suitable for unimodal problems.

2.2 Crossover operation

After the mutation operation, the DE algorithm uses a crossover operation, often referred to as discrete recombination, in which the mutated individual $\mathbf{V}_{i,gen}$ is mated with $\mathbf{X}_{i,gen}$ and generates the offspring $\mathbf{U}_{i,gen}$. The elements of an individual $\mathbf{U}_{i,gen}$ are inherited from $\mathbf{X}_{i,gen}$ and $\mathbf{V}_{i,gen}$, which are determined by the parameter crossover probability (CR $\in [0, 1]$), as follows:

$$\mathbf{U}_{i,d,gen} = \begin{cases} \mathbf{V}_{i,d,gen}, & \text{if rand}(d) \leq \text{CR} \\ \mathbf{X}_{i,d,gen}, & \text{if rand}(d) > \text{CR} \end{cases} \quad (7)$$

where $d = 1, 2, \ldots, D$ denotes the dth element of individual vectors, D is total number of elements of the individual vector and rand $(d) \in [0, 1]$ is the dth evaluation of a random number generator.

2.3 Selection operation

The DE algorithm applies the selection operation to determine whether the individual survives to the next generation. A knockout competition occurs between each individual $\mathbf{X}_{i,gen}$ and its offspring $\mathbf{U}_{i,gen}$, and the winner, selected deterministically based on objective function values, is then promoted to the next generation. The selection operation is described as follows:

$$\mathbf{X}_{i,gen+1} = \begin{cases} \mathbf{X}_{i,gen}, & \text{if fitness}(\mathbf{X}_{i,gen}) < \text{fitness}(\mathbf{U}_{i,gen}) \\ \mathbf{U}_{i,gen}, & \text{otherwise} \end{cases} \quad (8)$$

where fitness(z) is the fitness value of individual z. After the selection operation, the population gets better or retains the same fitness value, but never deteriorates.

3 Group-based differential evolution

This section describes the GDE learning process. This learning process groups the population into a superior group and an inferior group. The two groups perform different mutation operations to produce offspring for the next generation. An adaptive group size tuning strategy is also applied to find a suitable group size.

3.1 GDE algorithm

In the DE algorithm, the mutation operation is a kernel operator that acts on all individuals to search potential solutions and leads to successful evolution performance. For various problems, different mutation strategies are often employed in the DE algorithm. Choosing the correct mutation strategy to model a practical problem is difficult, however. Therefore, in this study, a GDE model is proposed with exploration and exploitation abilities. The model combines the two mutation strategies to enhance performance for solving practical problems. A flow chart of the GDE algorithm is shown in Fig. 2.

In the first step of the GDE, a population of *NP* *D*-dimensional individuals is generated by a uniformly random process, and the fitness value of all individuals is determined. A sorting process is used to arrange all individuals

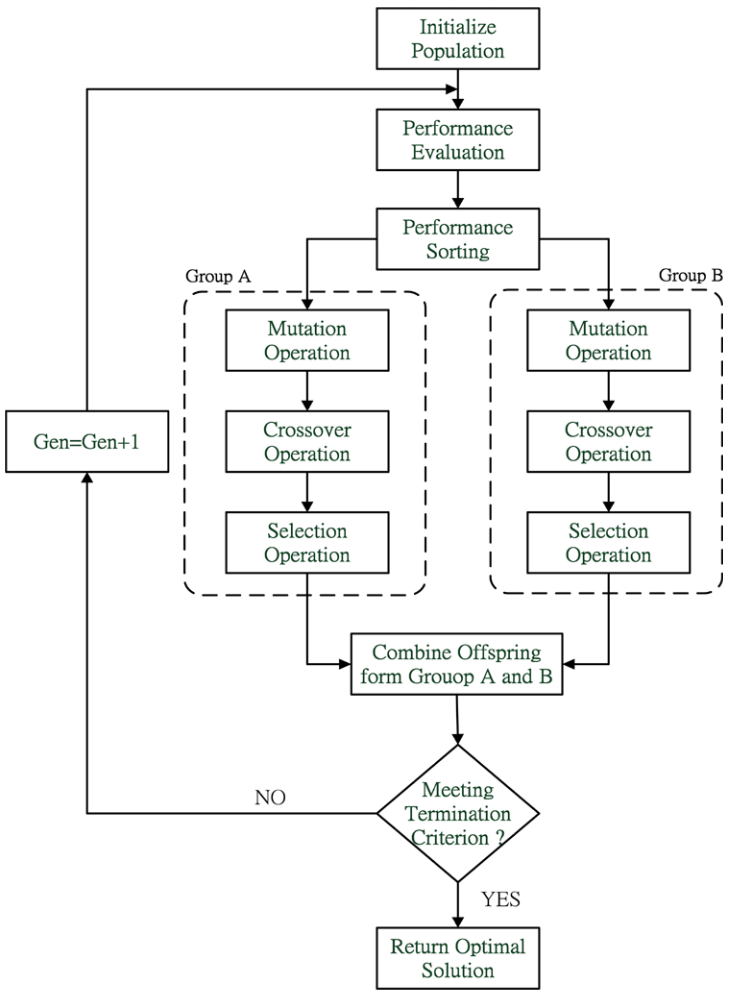


Fig. 2 The flow chart of GDE algorithm. *Gen* is the generation counter

based on increasing fitness, i.e., $fitness_1 < fitness_2 < \cdots < fitness_{NP-1} < fitness_{NP}$ for minimum objective problems. According to the fitness values, we group all individuals into an inferior group and a superior group, called Group A and Group B. Group A, containing *GS* individuals with the lowest fitness values, performs a global search to increase the diversity in the population and widely seek potential solutions. The other $(NP - GS)$ individuals composing Group B perform a local search to actively detect better solutions near their current best position. A complete mutation operation based on Group A and Group B is shown as follows:

$$\text{Group A}: \quad \mathbf{V}_{i,gen} = \mathbf{X}_{i,gen} + F_a(\mathbf{X}_{r1,gen} - \mathbf{X}_{r2,gen}), \quad i = 1, 2, \ldots, GS \tag{9}$$

$$\text{Group B} \quad \mathbf{V}_{j,gen} = \mathbf{X}_{gbest,gen} + F_b(\mathbf{X}_{r3,gen} - \mathbf{X}_{r4,gen}), \quad j = 1, 2, \ldots, (NP - GS) \tag{10}$$

where F_a and F_b are scale factors; GS indicates group size; $\mathbf{X}_{r1,gen}$, $\mathbf{X}_{r2,gen}$, $\mathbf{X}_{r3,gen}$ and $\mathbf{X}_{r4,gen}$ are random individuals selected from the population; and $i \neq r1 \neq r2$ and $r3 \neq r4$. Next, crossover and selection operations are performed, as shown in Sect. 2 for the traditional DE process. All steps are repeated until the process reaches the terminal condition.

3.2 Self-adaptive strategy based on the 1/5th rule

Parameter control, which can directly influence the convergence speed and search capability, is an important task in the evolutionary algorithm [39, 45]. Previously, the trial-and-error method for choosing suitable parameters was used, requiring multiple optimization runs. For this model, however, a self-adaptive approach is proposed based on the 1/5th rule for dynamic group size tuning. The 1/5th rule [46, 47] is a well-known method for parameter tuning and is usually used in evolution strategies for controlling the mutation strength. The idea of the 1/5th rule is to balance local searching and global searching based on the probability of success, P. If $P > 1/5$, then the algorithm increases the global search capability; otherwise, if $P > 1/5$, the algorithm increases the local search capability. If $P = 1/5$, the algorithm stays the same. Based on this concept, a group size with similar characteristics is adapted by the 1/5th rule to balance the exploration and exploitation abilities in GDE. A larger GS

Initialization: assign initial group size *GS*, adjustment factor $\beta \in [0,1]$ and period G_p as number of generation.

```
Algorithm:
1) Perform GDE algorithm.
2) Every $G_p$ generation,
    ○ Calculate the number $N_s$ of updating global best
      solution in Group B.
    ○ Determine an estimate of success probability $P$ as
```

$$P = \frac{N_s}{G_p} \tag{11}$$

```
    ○ Change group size $GS$ according to
```

$$GS = \begin{cases} round(GS/\beta) & \text{if } P > 1/5 \\ round(GS \cdot \beta) & \text{if } P < 1/5 \\ GS & \text{if } P = 1/5 \end{cases} \tag{12}$$

Fig. 3 1/5th rule algorithm for group size tuning in the GDE algorithm

means that the GDE favors exploration to increase the population diversity. A smaller GS means that the GDE favors exploitation to actively search for better solutions near the current best position. Therefore, the 1/5th rule is an appropriate parameter tuning strategy for the GDE. The complete self-adaptive parameter tuning strategy, based on the 1/5th rule, is shown in Fig. 3. In this process, the probability of success is calculated by counting the successful instances updating the best solution. The group size is adjusted by an adjustment factor β. In (17), *round*(.) is to set the value to the nearest integer.

When $GS = NP$, the mutation operation completely favors global information for evolving a better individual. In this case, the GDE algorithm equals the traditional DE algorithm with the DE/rand/1 mutation operation. At the other extreme, when $GS = 0$, the GDE algorithm completely favors local information, which equals the traditional DE algorithm with the *DE/best/1* mutation operation.

4 Simulation results

To verify the performance of the new algorithm, a set of thirteen classical benchmark test functions [14, 48, 49] is used for comparison. The analytical form of these functions is given in Table 1, where D denotes the dimensionality of the problem. Based on their properties, the functions can be divided into unimodal functions and multimodal functions. The functions f_1 through f_4 are continuous unimodal functions; f_5 is a discontinuous step function; f_6 is a noisy quartic function; f_7 is the Rosenbrock function; which is a multimodal function problem for $D > 3$ [48]; and f_8 through f_{13} are multimodal functions, with number of local minima increasing exponentially with the problem dimension [49]. In addition, f_8 is the only boundary-constrained function investigated in this study. All these functions have an optimal value at zero.

The GDE algorithm is compared with three classic DE algorithms, DE/rand/1, DE/best/1, and DE/target-to-best. For comparison, the parameters of the GDE algorithm are fixed at $F_a = 0.5$, $F_b = 0.8$, $CR_a = CR_b = 0.9$, initial group size $GS = NP/2$, adjustment factor $\beta = 0.9$ and $Gp = 20$ in all simulations. The parameter settings for the three classic DE algorithms are recommended and used as follows:

DE/rand/1: $F = 0.5$ and $CR = 0.9$ [22, 35, 50]

DE/best/1: $F = 0.8$ and $CR = 0.9$ [31]

DE/target-to-best/1: $F = 0.8$ and $CR = 0.9$ [33].

Many authors report success with these parameter settings. In all simulations, the population size, *NP*, is set to 100 and 300 in the cases of $D = 10$, and $D = 30$, respectively. The maximum number of function evaluations (FEs) is set to 100,000 when solving 10-D problems, and 300,000 when solving 30-D problems. All results are based on 50 independent runs. Section 4.3 demonstrates the significant difference between the methods based on a statistical comparison process. An additional simulation based on a two-difference vector is discussed in Sect. 4.4.

4.1 Results for the 10-D numerical function problem

In this simulation, the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms are applied to 10-dimensional problems on 13 benchmark test functions. Table 2 shows the detailed performance of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms, including the mean and STD performance for 50 independent runs. From this table, the proposed GDE achieves better performance than the other algorithms and obtains the best results for nine of the thirteen functions. When comparing only the three traditional DE algorithms, the DE/target-to-best/bin algorithm often performs better than either the DE/rand/bin or the DE/best/1 algorithm for the benchmark test functions. The DE/best/1 algorithm shows different behavior from the other three traditional DE algorithms when applied to f_{11}. The learning curves of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms for all 13 test functions for low dimensional ($D = 10$) problems are shown in Fig. 4. Based on the results, GDE converges faster than the other algorithms for all 13 benchmark test functions.

4.2 Results for the 30-D numerical function problem

To verify the capability of the algorithm on 30-dimensional problems, the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms are applied to the 13 benchmark test

Table 1 Benchmark functions. D is the dimensional of the function

Group	Test functions	Search range	D	Optimal value
Unimodal	$f_1=\sum_{i=1}^{D}(x_i)^2$	$[-100, 100]^D$	10 and 30	0
	$f_2=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	$[-10, 10]^D$		
	$f_3=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_i\right)^2$	$[-100, 100]^D$		
	$f_4=\max_i\lvert x_i\rvert$	$[-100, 100]^D$		
	$f_5=\sum_{i=1}^{D}(x_i+0.5)^2$	$[-100, 100]^D$		
	$f_6=\sum_{i=1}^{D}ix_i^4+rand[0,1)$	$[-1.28, 1.28]^D$		
Multimodal	$f_7=\sum_{i=1}^{D}\left[100\left(x_{i+1}-x_i^2\right)+(x_i-1)^2\right]$	$[-30, 30]^D$		
	$f_8=\sum_{i=1}^{D}-x_i\sin\sqrt{\lvert x_i\rvert}+D\cdot 418.98288727243369$	$[-500, 500]^D$		
	$f_9=\sum_{i=1}^{D}\left[x_i-10\cos(2\pi x_i)+10\right]$	$[-5.12, 5.12]^D$		
	$f_{10}=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)\right)+20+e$	$[-32, 32]^D$		
	$f_{11}=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600, 600]^D$		
	$f_{12}=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_D-1)^2\right\}+\sum_{i=1}^{D}u(x_i,10,100,4)$ where $y_i=1+\frac{1}{4}(x_i+1)$ and $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & \text{if } x_i>a\\ k(-x_i-a)^m, & \text{if } x_i<-a\\ 0, & \text{otherwise}\end{cases}$	$[-50, 50]^D$		
	$f_{13}=\frac{1}{10}\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_D-1)^2\left[1+\sin^2(2\pi x_D)\right]\right\}+\sum_{i=1}^{D}u(x_i,10,100,4)$ where $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & \text{if } x_i>a\\ k(-x_i-a)^m, & \text{if } x_i<-a\\ 0, & \text{otherwise}\end{cases}$	$[-50, 50]^D$		

Table 2 Experimental results of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms for 10 dimensional problems, averaged over 50 independent runs

Function	Max.FEs	GDE	DE/rand/1	DE/best/1	DE/target-to-best/1
		Mean (STD)			
f_1	100,000	**4.821e-46 (1.137e-45)**	1.382e-36 (1.193e-36)	1.602e-39 (1.392e-39)	2.954e-41 (2.694e-41)
f_2	100,000	**2.875e-21 (4.999e-21)**	7.475e-19 (3.587e-19)	4.291e-20 (3.637e-20)	9.352e-21 (3.646e-21)
f_3	100,000	**1.338e-24 (2.777e-24)**	1.168e-20 (9.579e-21)	1.084e-22 (1.351e-22)	4.685e-24 (3.772e-24)
f_4	100,000	1.042e-14 (2.914e-14)	3.048e-13 (2.354e-13)	2.683e-14 (2.802e-14)	**3.943e-15 (1.695e-15)**
f_5	100,000	1.325e-15 (3.371e-15)	2.325e-12 (3.350e-12)	2.278e-21 (3.287e-21)	**5.986e-22 (1.260e-22)**
f_6	100,000	**0.000e+00 (0.000e+00)**	**0.000e+00 (0.000e+00)**	**0.000e+00 (0.000e+00)**	**0.000e+00 (0.000e+00)**
f_7	100,000	**1.329e-03 (6.047e-04)**	1.78e-03 (6.776e-04)	2.011e-03 (8.390e-04)	1.694e-03 (7.761e-04)
f_8	100,000	**2.787e+02 (1.871e+02)**	2.460e+02 (3.611e+02)	6.664e+02 (3.734e+02)	5.012e+02 (1.347e+02)
f_9	100,000	**5.597e+00 (1.570e+00)**	1.882e+01 (3.235e+00)	6.467e+00 (1.641e+00)	2.192e+01 (3.391e+00)
f_{10}	100,000	7.993e-15 (2.901e-15)	**4.440e-15 (0.000e+00)**	5.151e-15 (1.497e-15)	**4.440e-15 (0.000e+00)**
f_{11}	100,000	8.883e-02 (4.691e-02)	1.819e-02 (9.154e-02)	**1.219e-02 (1.021e-01)**	3.127e-02 (8.615e-02)
f_{12}	100,000	**4.7116e-32 (1.153e-47)**	**4.7116e-32 (1.153e-47)**	**4.711e-32 (1.153e-47)**	**4.711e-32 (1.153e-47)**
f_{13}	100,000	**1.349e-32 (2.884e-48)**	**1.349e-32 (2.884e-48)**	**1.349e-32 (2.884e-48)**	**1.349e-32 (2.884e-48)**
Average Rank		9/13	4/13	4/13	6/13

functions. Table 3 shows the detailed performance of the GDE, DE/rand/bin, DE/best/bin and DE/target-to-best/bin algorithms, including the means and STD performance for 50 independent runs. As shown in the table, the GDE algorithm performs better than the other algorithms on the 13 benchmark test functions. In 30-dimensional problems, the traditional DE algorithms have problems obtaining better solutions. The GDE algorithm results in the best solution for thirteen out of thirteen functions. Comparing only the three traditional DE algorithms, the DE/target-to-best/1 algorithm shows very different behavior when applied to f_4 and f_{12}. The DE/target-to-best/1 algorithm performs better overall from among the three traditional DE algorithms. The learning curves of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms for the 13 test functions applied to the 30-dimensional problems are shown in Fig. 5. This figure shows that GDE converges faster than the other algorithms on both unimodal and multimodal function problems.

4.3 Statistical comparison using the Wilcoxon signed ranks test

To understand the significant difference between the GDE and the traditional DE algorithms applied to multiple test functions, a statistical procedure based on the Wilcoxon signed ranks test [51, 52] is performed. This test, a nonparametric alternative to the paired t-test, ranks the differences in performance of two models for each data set, ignoring the signs, and then compares the ranks for the positive and the negative differences. In this study, the GDE is chosen as the control algorithm to compare with the traditional DE algorithms. The performance of an algorithm differs significantly if the corresponding statistic z differs by at least

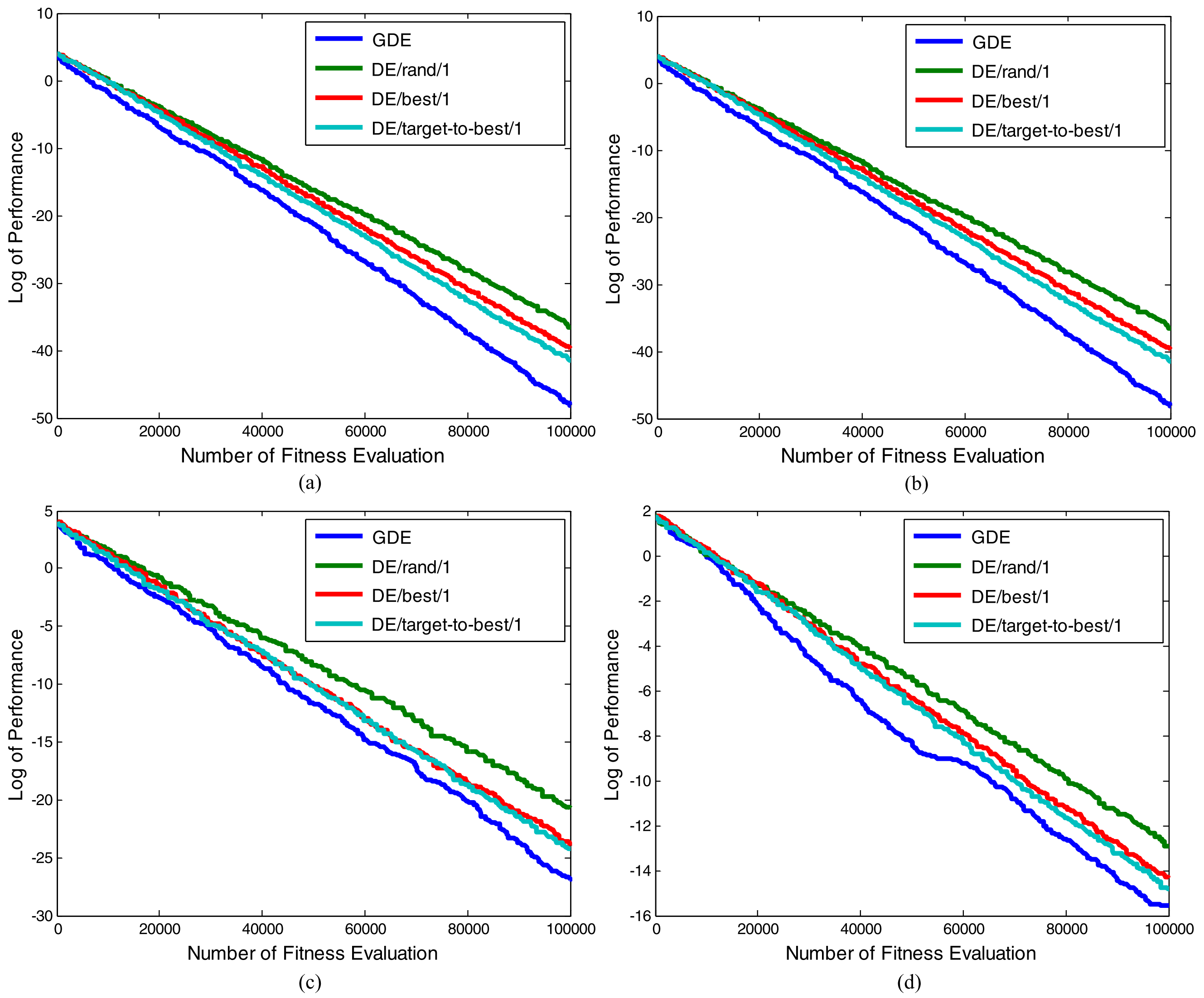


Fig. 4 The best learning curve of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms on 13 test function for 10 dimensional problems. (**a**) Function 1: f_1; (**b**) Function 2: f_2; (**c**) Function 3: f_3; (**d**) Function 4: f_4; (**e**) Function 5: f_5; (**f**) Function 6: f_6; (**g**) Function 7: f_7; (**h**) Function 8: f_8; (**i**) Function 9: f_9; (**j**) Function 10: f_{10}; (**k**) Function 11: f_{11}; (**l**) Function 12: f_{12}; (**m**) Function 13: f_{13}

the critical value, -1.96. The statistic z is calculated as follows:

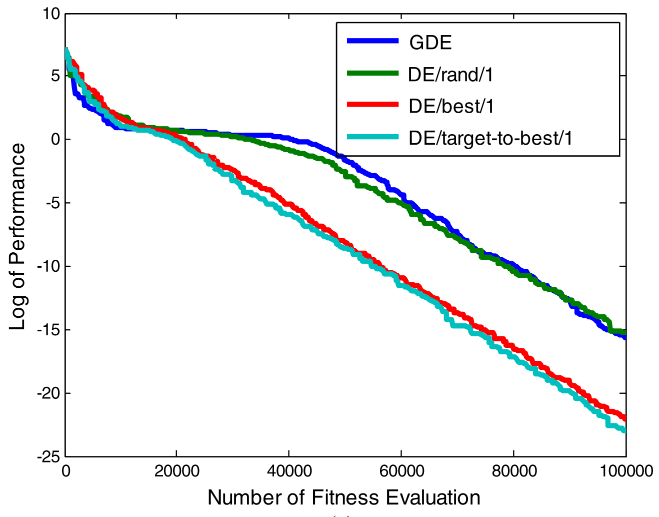
$$z = \frac{\mathrm{Min}(R+, R-) - \frac{1}{4}N(N+1)}{\sqrt{\frac{1}{24}N(N+1)(2N+1)}}, \tag{11}$$

where N is the number of test functions, $R+$ is the sum of ranks for the data sets on which the second algorithm outperformed the first, and $R-$ is the sum of ranks for the opposite case.

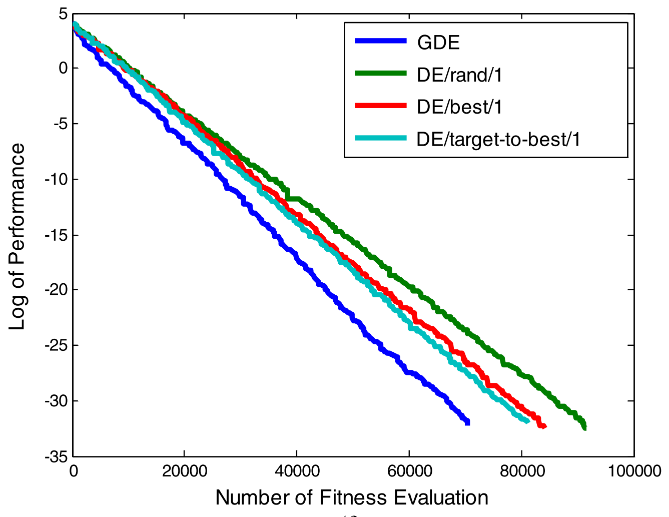
Statistical comparisons are performed for both 10- and 30-dimensional problems and $N = 26$. Table 4 presents a complete set of results for the Wilcoxon signed ranks test. The statistic $z = -2.73, -3.50$ and -2.82 for GDE versus DE/rand/1, GDE versus DE/best/1 and GDE versus DE/target-to-best/1, respectively. For all cases, the statistic z is smaller than the critical value, which means that the GDE is significantly better than DE/rand/1, DE/best/1 and DE/target-to-best/1 in this simulation.

4.4 Comparison with other algorithms

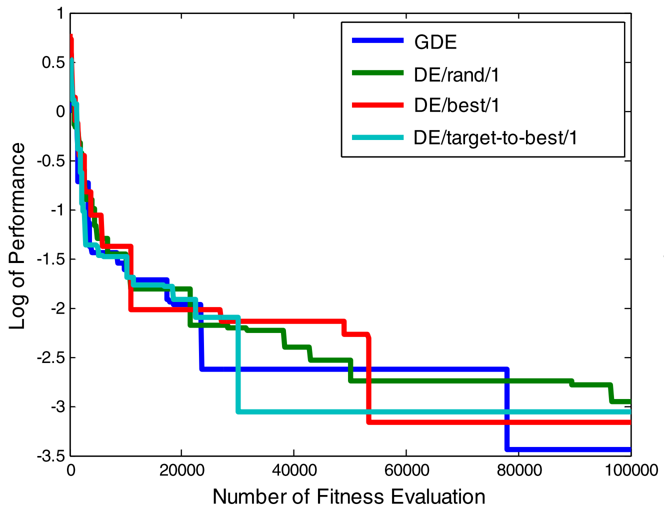
Further results regarding the comparison of the GDE algorithm with other evolutionary algorithms is presented in this section. These algorithms include CEP [53], ALEP [54], BestLevy [54], NSDE [55] and RTEP [53]. Table 5 shows the comparative results with respect to 30-dimensional problems. The GDE algorithm showed the best results for five of eight functions, i.e., Function 1, Function 3, Function 10, Function 12 and Function 13. The overall results show that



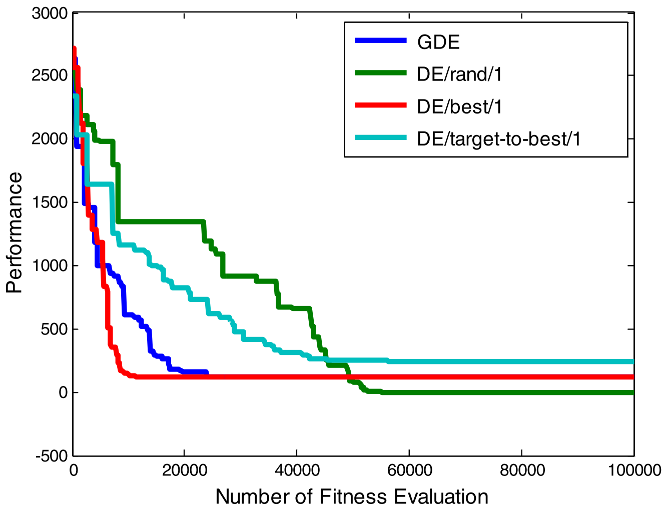
(e)



(f)



(g)



(h)

Fig. 4 (*Continued*)

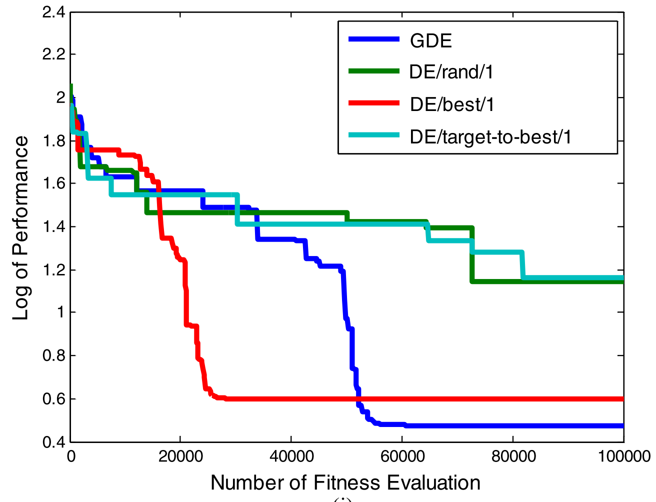
the GDE algorithm is a more effective algorithm than other competitive algorithms.

5 Conclusions

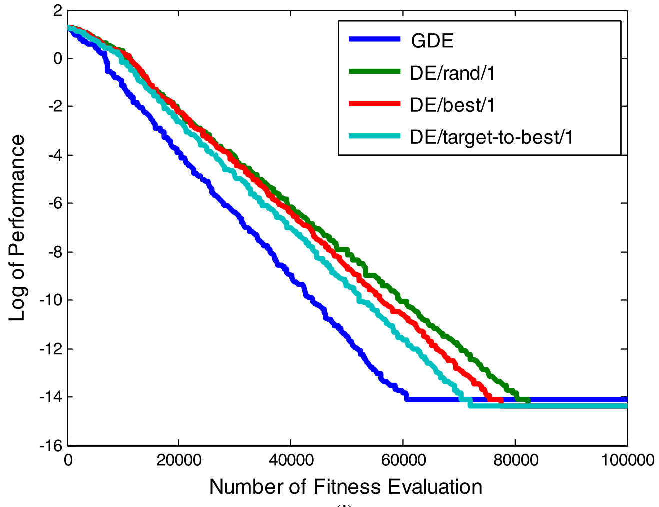
This study has proposed a group-based DE algorithm for numerical optimization problems. The GDE algorithm performs two mutation operations based on different groupings to effectively search for the optimal solution. This algorithm, which has both exploitation and exploration abilities, is a generalized DE model. In addition, an adaptive parameter tuning strategy based on the 1/5th rule is used to dynamically adjust the group size. The simulation results demonstrate that the GDE method performs better than other EAs for optimization problems.

Two advanced topics on the proposed GDE should be addressed in future research. First, the GDE may adopt other further learning methods to improve performance. For example, Norouzzadeh [17] used a fuzzy logic to enhance performance in PSO. This method increases the possibilities when searching potential solutions. Additionally, future simulations will include applying the GDE to neuro-fuzzy system optimizations.

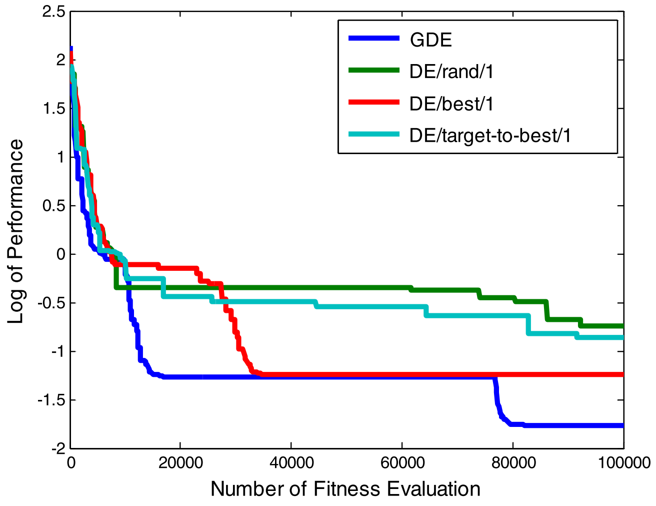
Acknowledgements This work was supported by Department of Industrial Technology under grants 100-EC-17-A-02-S1-032, by the UST-UCSD International Center of Excellence in Advanced Bioengineering sponsored by the Taiwan National Science Council I-RiCE Program under Grant Number NSC-100-2911-I-009-101 and by the Aiming for the Top University Plan of National Chiao Tung University, the Ministry of Education, Taiwan, under Contract 100W9633 & 101W963.



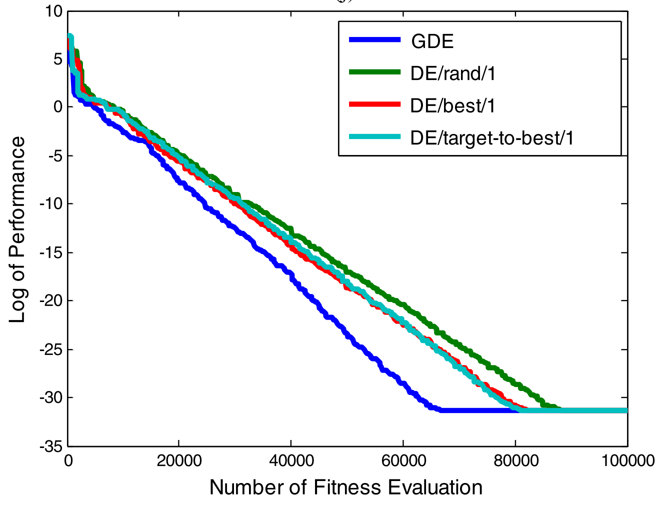
(i)



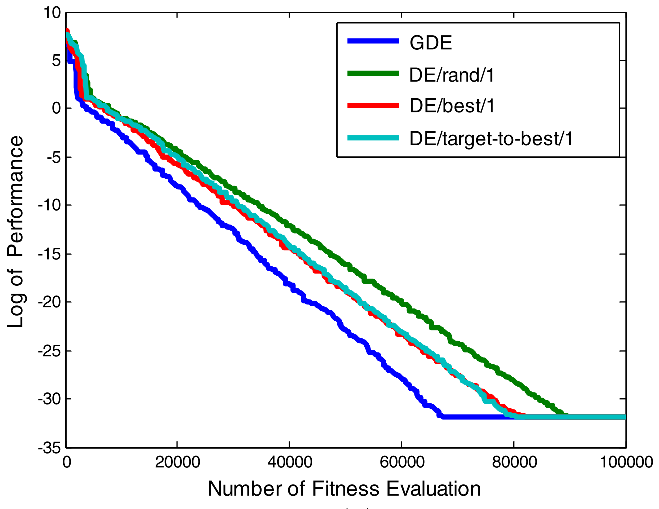
(j)



(k)



(l)



(m)

Fig. 4 (*Continued*)

Table 3 Experimental results of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms for 30 dimensional problems, averaged over 50 independent runs

Function	Max.FEs	GDE	DE/rand/1	DE/best/1	DE/target-to-best/1
		Mean (STD)			
f_1	300,000	**6.074e-24 (8.536e-24)**	1.355e-03 (5.304e-04)	4.968e-04 (3.323e-04)	1.096e-04 (4.7257e-05)
f_2	300,000	**1.759e-07 (4.185e-07)**	2.130e-01 (7.311e-02)	2.882e-02 (7.528e-03)	2.040e-02 (8.340e-03)
f_3	300,000	**1.746e-02 (2.105e-02)**	1.314e+03 (3.752e+02)	4.742e+02 (1.814e+02)	2.692e+02 (7.346+01)
f_4	300,000	**3.256e-01 (2.675e-01)**	2.813e+00 (3.646e+01)	1.000e+00 (3.224e-01)	8.337e-01 (1.919e-01)
f_5	300,000	**5.217e+00 (5.189e+00)**	2.722e+01 (6.322e-01)	3.384e+01 (2.827e+01)	2.856e+01 (2.035e+01)
f_6	300,000	**1.557e-23 (2.651e-23)**	1.363e-03 (3.836e-04)	6.735e-04 (3.156e-04)	1.032e-04 (3.624e-05)
f_7	300,000	**1.899e-02 (6.103e-03)**	2.483e-02 (6.148e-03)	2.759e-02 (6.852e-03)	2.029e-02 (5.103e-03)
f_8	300,000	**2.897e+03 (8.860e+02)**	7.000e+03 (2.866e+02)	3.097e+03 (7.152e+02)	4.377e+03 (1.338e+03)
f_9	300,000	**4.745e+01 (1.201e+01)**	1.964e+02 (7.629e+01)	1.106e+02 (1.898e+01)	2.019e+02 (6.946e+00)
f_{10}	300,000	**2.129e-10 (1.127e-10)**	1.796e-02 (3.406e-03)	8.160e-03 (2.819e-03)	3.603e-03 (9.845e-04)
f_{11}	300,000	**8.127e-03 (9.785e-03)**	7.260e-03 (2.931e-03)	5.785e-03 (5.361e-03)	4.030e-03 (3.991e-03)
f_{12}	300,000	**6.133e-21 (7.051e-22)**	5.678e-04 (2.638e-04)	1.191e-04 (7.364e-05)	3.317e-05 (3.053e-05)
f_{13}	300,000	**5.541e-23 (9.190e-23)**	2.508e-03 (9.607e-04)	1.401e-03 (3.463e-03)	1.024e-04 (7.882e-05)
Average Rank		13/13	0/13	0/13	0/13

Table 4 Result of Wilcoxon signed ranks test for numerical function problems

Algorithm	Min($R+$, $R-$)	z	Critical value	Final result
DE/rand/1	48	−2.73	−1.96	Rejected the hypothesis
DE/best/1	43	−3.50		Rejected the hypothesis
DE/target-to-best/1	52	−2.82		Rejected the hypothesis

Table 5 Comparison with other evolutionary algorithms ($D = 30$), including GDE, CEP [53], ALEP [54], BestLevy [54], NSDE [55] and RTEP [53]

Function	GDE	CEP [53]	ALEP [54]	BestLevy [54]	NSDE [55]	RTEP [53]
	Mean					
f_1	**6.07e-24**	9.10e-04	6.32e-04	6.59e-04	7.10e-17	7.50e-18
f_3	**1.74e-17**	2.10e+02	4.18e-02	3.06e+01	7.90e-16	2.40e-15
f_7	1.89e-02	8.60e+01	4.34e+01	5.77e+01	**5.90e-28**	1.10e+00
f_9	4.74e+01	4.34e+01	5.85e+00	1.30e+01	–	**2.50e-14**
f_{10}	**1.62e-10**	1.50e+00	1.90e-02	3.10e-02	1.69e-09	2.00e-10
f_{11}	8.12e-03	8.70e-00	2.4e-02	1.80e-02	5.80e-16	**2.70e-25**
f_{12}	**6.13e-21**	4.80e-01	6.00e-06	3.00e-05	5.40e-16	3.20e-13
f_{13}	**5.54e-23**	8.90e-02	9.80e-05	2.60e-04	6.40e-17	7.10e-08

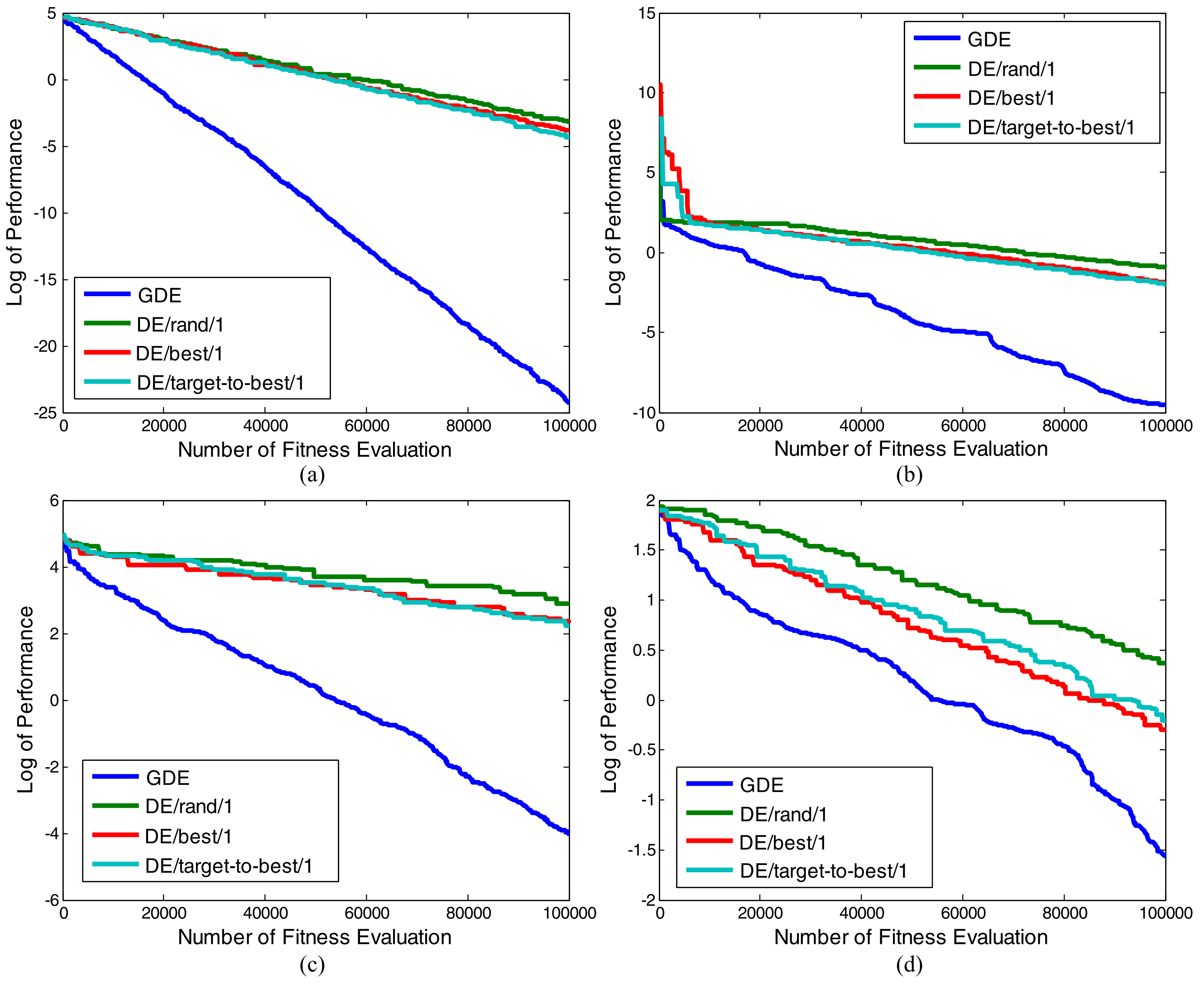
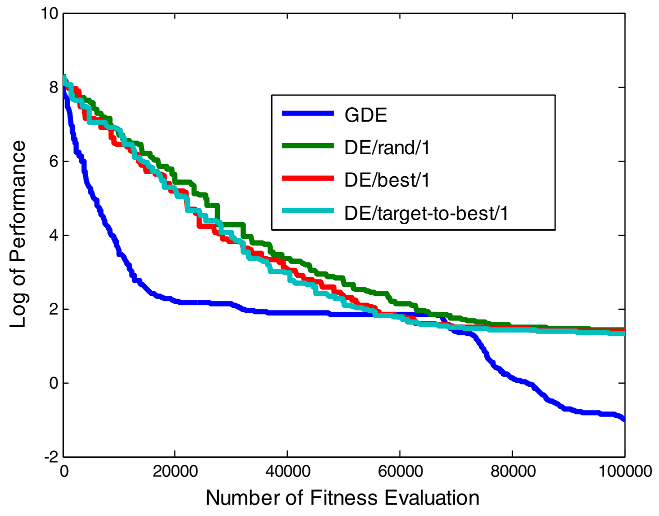
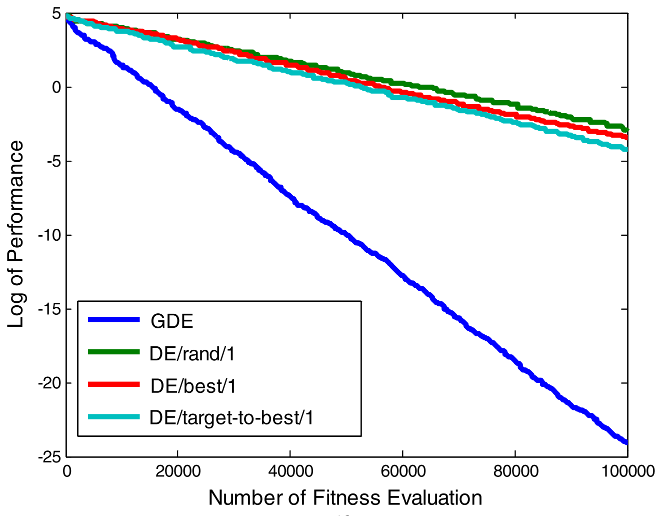


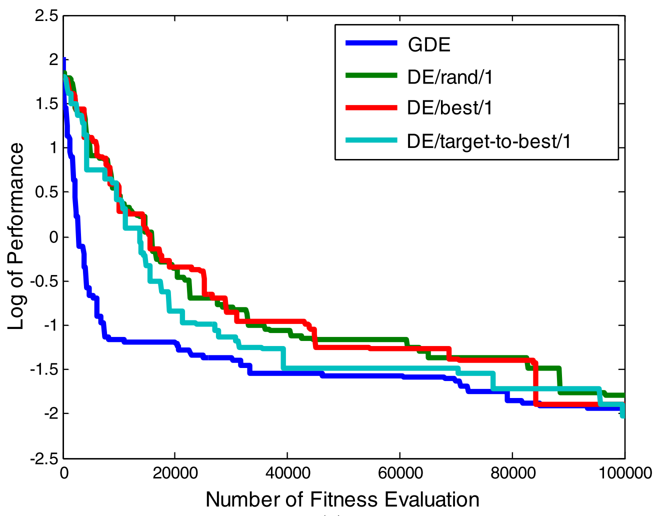
Fig. 5 The best learning curve of the GDE, DE/rand/1, DE/best/1 and DE/target-to-best/1 algorithms on 13 test functions for 30 dimensional problems. (**a**) Function 1: f_1; (**b**) Function 2: f_2; (**c**) Function 3: f_3; (**d**) Function 4: f_4; (**e**) Function 5: f_5; (**f**) Function 6: f_6; (**g**) Function 7: f_7; (**h**) Function 8: f_8; (**i**) Function 9: f_9; (**j**) Function 10: f_{10}; (**k**) Function 11: f_{11}; (**l**) Function 12: f_{12}; (**m**) Function 13: f_{13}



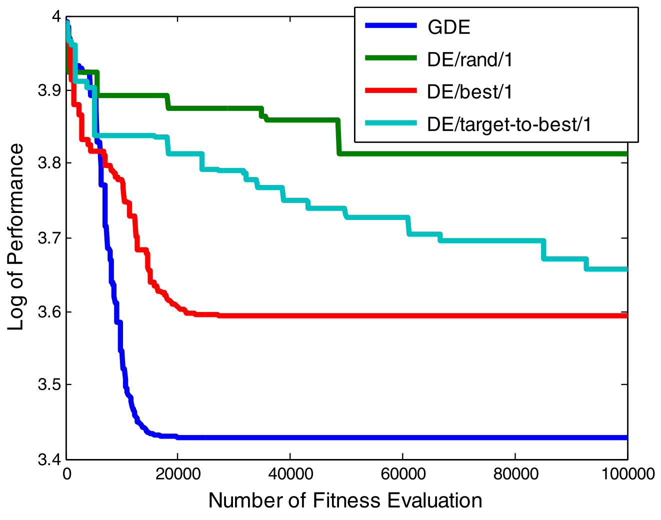
(e)



(f)

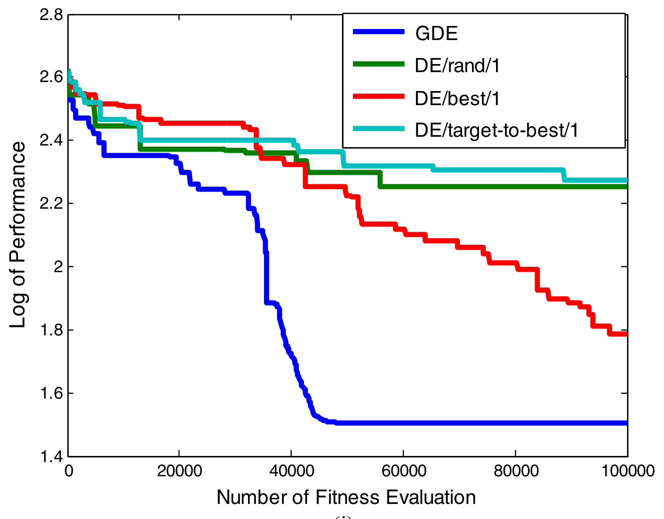


(g)

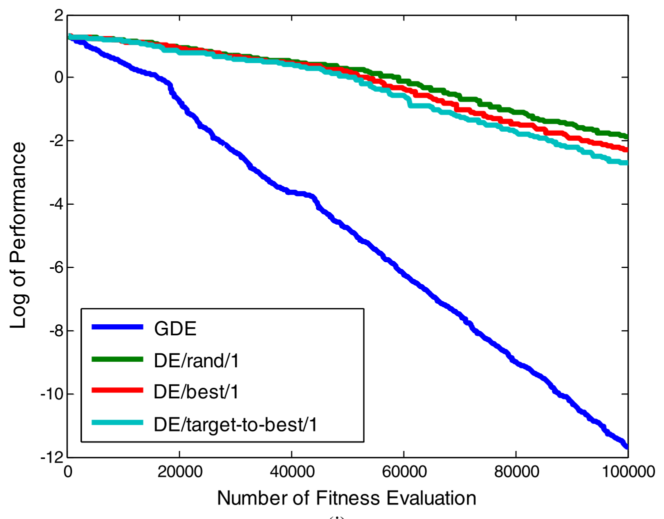


(h)

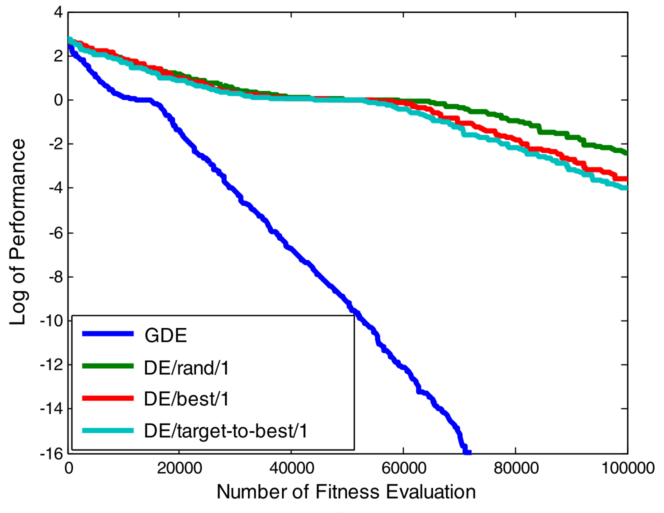
Fig. 5 (*Continued*)



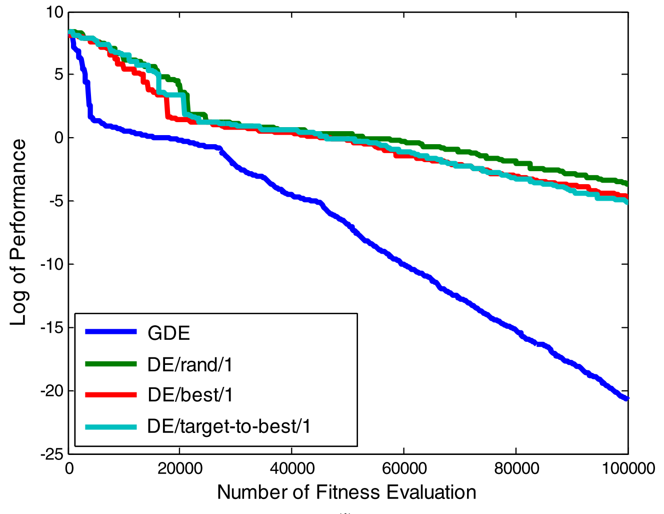
(i)



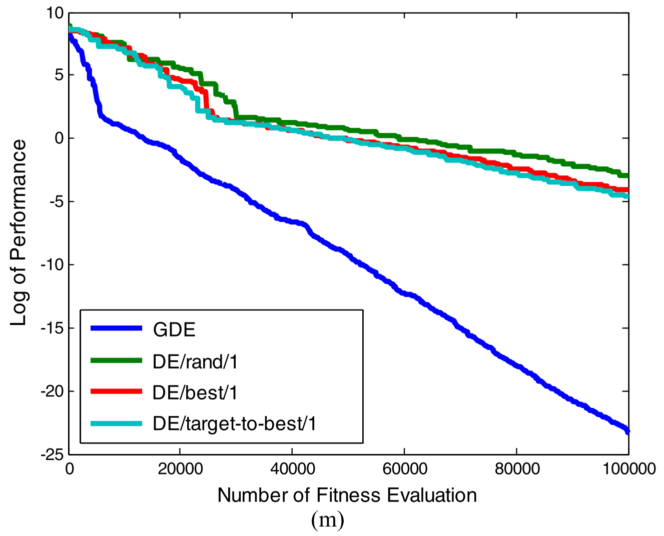
(j)



(k)



(l)



(m)

Fig. 5 (*Continued*)

References

1. Carlos CCA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191:1245–1287
2. Fei P, Ke T, Guoliang C, Xin Y (2010) Population-based algorithm portfolios for numerical optimization. IEEE Trans Evol Comput 14:782–800
3. Fogel DB (1995) Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, New York
4. Salto C, Alba E (2012) Designing heterogeneous distributed GAs by efficiently self-adapting the migration period. Appl Intell 36(4):800–808
5. Gacto MJ, Alcalá R, Herrera F (2012) A multi-objective evolutionary algorithm for an effective tuning of fuzzy logic controllers in heating, ventilating and air conditioning systems. Appl Intell 36(2):330–347
6. Shin KS, Jeong Y-S, Jeong MK (2012) A two-leveled symbiotic evolutionary algorithm for clustering problems. Appl Intell 36(4):788–799
7. Ayvaz D, Topcuoglu HR, Gurgen F (2012) Performance evaluation of evolutionary heuristics in dynamic environments. Appl Intell 37(1):130–144
8. Korkmaz EE (2010) Multi-objective genetic algorithms for grouping problems. Appl Intell 33(2):179–192
9. Chu C-P, Chang Y-C, Tsai C-C (2011) PC2PSO: personalized e-course composition based on particle swarm optimization. Appl Intell 34(1):141–154
10. Ali YMB (2012) Psychological model of particle swarm optimization based multiple emotions. Appl Intell 36(3):649–663
11. Wang K, Zheng YJ (2012) A new particle swarm optimization algorithm for fuzzy optimization of armored vehicle scheme design. Appl Intell. doi:10.1007/s10489-012-0345-0
12. Xing H, Qu R (2012) A compact genetic algorithm for the network coding based resource minimization problem. Appl Intell 36(4):809–823
13. Özyer T, Zhang M, Alhajj R (2011) Integrating multi-objective genetic algorithm based clustering and data partitioning for skyline computation. Appl Intell 35(1):110–122
14. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3:82–102
15. Bäck TT, Schwefel H-P (2002) Evolution strategies: a comprehensive introduction. Natural Computing, 3–52
16. Kennedy J, Eberhart R (1995) Particle swarm optimization. Paper presented at the IEEE int neural netw
17. Norouzzadeh MS, Ahmadzadeh MR, Palhang M (2012) LADPSO: using fuzzy logic to conduct PSO algorithm. Appl Intell 37(2):290–304
18. Ali YMB (2012) Psychological model of particle swarm optimization based multiple emotions. Appl Intell 36(3):649–663
19. Shuang B, Chen J, Li Z (2011) Study on hybrid PS-ACO algorithm. Appl Intell 34(1):64–73
20. Masoud H, Jalili S, Hasheminejad SMH (2012) Dynamic clustering using combinatorial particle swarm optimization. Appl Intell. doi:10.1007/s10489-012-0373-9
21. Price K, Storn R, Lampinen J (2005) Differential evolution: a practical approach to global optimization. Springer, Berlin
22. Storn R, Price K (1997) Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11:341–359
23. Salto C, Alba E (2012) Designing heterogeneous distributed GAs by efficiently self-adapting the migration period. Appl Intell 36(4):800–808
24. Araújo AFR, Garrozi C (2010) MulRoGA: a multicast routing genetic algorithm approach considering multiple objectives. Appl Intell 32(3):330–345
25. Lin C-T, Han M-F, Lin Y-Y, Liao S-H, Chang J-Y (2011) Neuro-fuzzy system design using differential evolution with local information. In: 2011 IEEE international conference on fuzzy systems (FUZZ), pp 1003–1006
26. Junhong L, Jouni L (2002) A fuzzy adaptive differential evolution algorithm. In: TENCON '02. Proceedings. 2002 IEEE region 10 conference on computers, communications, control and power engineering, vol 1, pp 606–611
27. Xue F, Sanderson AC, Bonissone PP, Graves RJ (2005) Fuzzy logic controlled multiobjective differential evolution. Paper presented at the IEEE int conf fuzzy syst
28. Brest J, Maučec MS (2008) Population size reduction for the differential evolution algorithm. Appl Intell 29(3):228–247
29. Cai Z, Gong W, Ling CX, Zhang H (2011) A clustering-based differential evolution for global optimization. Appl Soft Comput 11:1363–1379
30. Chen C-H, Lin C-J, Lin C-T (2009) Nonlinear system control using adaptive neural fuzzy networks based on a modified differential evolution. IEEE Trans Syst Man Cybern, Part C, Appl Rev 39:459–473
31. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. IEEE Trans Evol Comput 13:526–553
32. Das S, Suganthan PN (2011) Differential evolution: a survey of the state-of-the-art. IEEE Trans Evol Comput 15:4–31
33. Cheshmehgaz HR, Desa MI, Wibowo A (2012) Effective local evolutionary searches distributed on an island model solving bi-objective optimization problems. Appl Intell. doi:10.1007/s10489-012-0375-7
34. Vafashoar R, Meybodi MR, Momeni Azandaryani AH (2012) CLA-DE: a hybrid model based on cellular learning automata for numerical optimization. Appl Intell 36(3):735–748
35. Jingqiao Z, Sanderson AC (2009) JADE: adaptive differential evolution with optional external archive. IEEE Trans Evol Comput 13:945–958
36. Mezura-Montes E, Miranda-Varela ME, del Carmen Gmez-Ramn R (2010) Differential evolution in constrained numerical optimization: an empirical study. Inf Sci 180:4223–4262
37. Noman N, Iba H (2008) Accelerating differential evolution using an adaptive local search. IEEE Trans Evol Comput 12:107–125
38. Qin AK, Huang VL, Suganthan PN (2009) Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 13:398–417
39. Qin AK, Suganthan PN (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: The 2005 IEEE congress on evolutionary computation, 2005, vol 2, pp 1785–1791
40. Rahnamayan S, Tizhoosh HR, Salama MMA (2008) Opposition-based differential evolution. IEEE Trans Evol Comput 12:64–79
41. Su M-T, Chen C-H, Lin C-J, Lin C-T (2011) A rule-based symbiotic modified differential evolution for self-organizing neuro-fuzzy systems. Appl Soft Comput 11:4847–4858
42. Wenyin G, Zhihua C, Ling CX, Hui L (2011) Enhanced differential evolution with adaptive strategies for numerical optimization. IEEE Trans Syst Man Cybern, Part B, Cybern 41:397–413
43. Vesterstrom J, Thomsen R (2004) A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In: Congress on evolutionary computation, 2004 (CEC2004), vol 2, pp 1980–1987

44. Lin CT, Han MF, Lin YY, Chang JY, Ko LW (2010) Differential evolution based optimization of locally recurrent neuro-fuzzy system for dynamic system identification. Paper presented at the 17th national conference on fuzzy theory and its applications
45. Josef T (2009) Adaptation in differential evolution: a numerical comparison. Appl Soft Comput 9:1149–1155
46. Bäck TT, Schwefel H-P (1995) Evolution strategies I: variants and their computational implementation. In: Genetic algorithms in engineering and computer science, pp 111–126
47. Beyer HG, Schwefel HP (2002) Evolution strategies: a comprehensive introduction. Nat Comput 3–52
48. Shang Y-W, Qiu Y-H (2006) A note on the extended Rosenbrock function. Evol Comput 14:119–126
49. Yao X, Liu Y, Liang K-H, Lin G (2003) Fast evolutionary algorithms. Paper presented at the advances evol computing: theory applicat, New York
50. Brest J, Greiner S, Boskovic B, Mernik M, Zumer V (2006) Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans Evol Comput 10:646–657
51. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
52. García S, Herrera F (2008) An extension on “Statistical comparisons of classifiers over multiple data sets” for all pairwise comparisons. J Mach Learn Res 9:2677–2694
53. Alam MS, Islam MM, Xin Yao F, Murase K (2011) Recurring two-stage evolutionary programming: a novel approach for numeric optimization. IEEE Trans Syst Man Cybern, Part B, Cybern 41:1352–1365
54. Lee C, Yao X (2004) Evolutionary programming using mutations based on the Lévy probability distribution. IEEE Trans Evol Comput 8:1–13
55. Yang Z, He J, Yao X (2007) Making a difference to differential evolution. In: Advances metaheuristics hard optimization, pp 397–414

Ming-Feng Han received the M.S. degree in electrical engineering from the National Central University, Taoyuan, Taiwan, R.O.C., in 2008. He is currently working toward the Ph.D. degree in the Department of Electrical Control Engineering form National Chiao Tung University, Hsinchu, Taiwan, R.O.C. His current research interests are evolutionary algorithm, machine learning, neuro-fuzzy system design and optimization techniques.

Shih-Hui Liao received the B.S. degree from the Department of Mechatronics Engineering, Changhua University of Education, Changhua, Taiwan, in 2007 and the M.S. degree from the Department of Electrical Engineering, National Sun Yat-Sen University, Kaohsiung, Taiwan, in 2009. She is currently working toward the Ph.D. degree in Electrical and Control Engineering at National Chiao-Tung University, Hsinchu, Taiwan. Her research interests include machine learning, soft computing, and fuzzy systems.

Jyh-Yeong Chang received the M.S. degree in electronic engineering in 1980, from National Chiao-Tung University, Hsinchu, Taiwan, R.O.C., and the Ph.D. degree in electrical engineering from North Carolina State University, Raleigh, in 1987. During 1976–1978 and 1980–1982, he was a Research Fellow at Chung Shan Institute of Science and Technology, Lung-Tan, Taiwan. He is a Professor in the Department of Electrical and Control Engineering. His current research interests include neural fuzzy systems, video processing and surveillance, and bioinformatics.

Chin-Teng Lin received the B.S. degree in control engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1986, and the M.Sc. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1989 and 1992, respectively. He is currently the Chair Professor of electrical and computer engineering with NCTU. His research interests include biologically inspired information systems, neural networks, fuzzy systems, multimedia hardware/software, and cognitive neuroengineering. Dr. Lin was an Associate Editor of the IEEE Transactions on Systems, Man, and Cybernetics, Part II. He currently serves as the EIC of the IEEE Transactions on Fuzzy Systems.